Learning Generative Adversarial Networks from Multiple Data Sources

Trung Le[1] , **Quan Hoang**[1] , **Hung Vu**[2] , **Tu Dinh Nguyen**[1] , **Hung Bui**[3] and **Dinh Phung**[1]
[1]Faculty of Infomation Technology, Monash University, Australia
[2]AI Research Lab, Trusting Social, Australia
[3]Google DeepMind
{trunglm, quan.hoang, tu.dinh.nguyen, dinh.phung}@monash.edu, hung.vu@trustingsocial.com, buih@google.com

Abstract

Generative Adversarial Networks (GANs) are a powerful class of deep generative models. In this paper, we extend GAN to the problem of generating data that are not only close to a primary data source but also required to be different from auxiliary data sources. For this problem, we enrich both GAN's formulations and applications by introducing pushing forces that thrust generated samples away from given auxiliary data sources. We term our method *Push-and-Pull GAN* (P2GAN). We conduct extensive experiments to demonstrate the merit of P2GAN in two applications: generating data with constraints and addressing the mode collapsing problem. We use CIFAR-10, STL-10, and ImageNet datasets and compute Fréchet Inception Distance to evaluate P2GAN's effectiveness in addressing the mode collapsing problem. The results show that P2GAN outperforms the state-of-the-art baselines. For the problem of generating data with constraints, we show that P2GAN can successfully avoid generating specific features such as black hair.

1 Introduction

Modern machine learning systems need to deal with complex high-dimensional objects such as natural images, motion pictures, speeches, dialog texts and hand-written cursive drawings to name a few. Recent deep generative models, in particular Generative Adversarial Networks (GANs) [Goodfellow *et al.*, 2014] have quickly become a building block for designing powerful models to work with such high-dimensional objects. The idea of GAN is to train a generator $G(z)$ where z might come from any arbitrary distribution such that the distribution P_G induced over the values of $G(z)$ (s) as z varies is close to the true data distribution P_{data}. Once trained, generating a new sample is extremely efficient as one can simply draw z then feed it through $G(z)$ where $G(z)$ is a deep neural network (NN). Despite its simplicity, GAN has shown an enormous capacity in dealing with high-dimensional objects and has been enjoying remarkable success from image, video generation [Mathieu *et al.*, 2015], image-to-image translation [Isola *et al.*, 2017] to name a few [Goodfellow, 2016]. ,

However, GAN comes with some important limitations. Central to its formulation [Goodfellow *et al.*, 2014] is a minimax optimization problem whose Nash equilibrium point minimizes the Jensen-Shanon (JS) divergence between P_{data} and P_G. This JS divergence might be viewed as a 'pulling force' to move generated samples toward data samples. Other variants of GAN have extended this mechanism to different divergences, notably f-divergence family proposed in [Nowozin *et al.*, 2016] which generalizes JS divergence via a variational bound whose solution can be characterized tractably. Nonetheless, the ill-posedness of GAN minimax problem and the nature of f-divergence pose the inherent mode collapsing problem where generated samples tend to 'collapse' to a few modes, hence hindering the diversity of generating process [Goodfellow *et al.*, 2014; Goodfellow, 2016; Le *et al.*, 2018]. Overcoming this problem has become one of the main research themes in GAN with reasonable success, but still an open problem. Besides f-divergence, WGAN [Arjovsky *et al.*, 2017; Gulrajani *et al.*, 2017; Dam *et al.*, 2019] employs the Wasserstein distance and formulates the optimization through the Kantorovich duality, but has its own problems in training due to the constraints encountered in the optimization formulation.

From a practicality viewpoint, while enjoying its research success, GAN is still limited in exploiting data from multiple sources. One particular open and important problem is to extend its current setting to move beyond a single data source to work with multiple data distributions, which currently receives very little research attention. For example, one might generate realistic photos of the beach, but at the same time purposely avoid generating images of a storm whose data collection are available for both beach and storm scenes; or in abnormality detection where one not only has access to normal data, but also partially abnormal data to train with. Real-world scenarios like these have abundant applications and are open to explore. Our main contribution of this paper is to propose a novel approach, leveraging on the success of GAN and recent techniques for this open problem.

Specifically, assume that there exists multiple data distribution with P be the primary and m other auxiliary distributions $P_1, P_2, \ldots, P_m$. Our goal is to recover P via a gen-

This work was partially supported by the Australian Research Council (ARC) DP160109394.

erative distribution Q as *close* to P as possible (i.e., the pull force), but at the same time, being as *different* from all other P_i(s) as possible (i.e., the push force). It is important to note that Q will *not* be estimated explicitly in a parametric form, but instead a generator function G will be learned such that Q implicitly represents the induced distribution for $G(z)$ (s) where z comes from any arbitrary distribution. This can be then formulated as an optimization problem as in Eq. (1) under a generalized extension for $f-$divergence in the existence of multiple data distributions. Subsequently, we extend the theoretical results in [Nowozin *et al.*, 2016], showing that it is still possible to obtain a tractable solutions for the Q's generator, G, and efficient algorithms to train G as well the discriminators. Our proposed model naturally subsumes and extends several important existing variants of GAN, including the original GAN [Goodfellow *et al.*, 2014], f-GAN [Nowozin *et al.*, 2016], and D2GAN [Nguyen *et al.*, 2017]. In addition, unlike GAN, the our discriminators at convergence point do not become uniform and redundant, but carry specific meanings which can be exploited for various application uses.

Beside the model contribution, while potentially having a wider application scope, we choose to apply and demonstrate our approach for two specific applications in this paper:

1. *Improving GAN training by overcoming mode collapsing problem*. Interestingly, we show that our approach is flexible enough to be exploited to improve the training of GAN (with a single data distribution), addressing the mode collapsing problem mentioned earlier. To do so, we train the first generator G_1 so that its induced distribution is close to P_{data} as usual. G_1 is anticipated to cover only some modes in P_{data} and known to suffer from the mode collapse problem. To subsequently diversify the generated samples, we need the generator to explore uncovered modes from G_1. Using the proposed approach, we then train the second generator G_2 which is to be as close to P_{data} as possible, but as different from G_1 as possible. This process is repeated until the generators cover sufficient number of modes, resulting in a sequence of generators $\{G_1, G_2, \ldots, G_k\}$. At the generation step, for each sample to be generated, we simply pick a random generator from this pool.

2. *Generating samples with constraints*. This is an ongoing research problem which has been addressed under both supervised and unsupervised setting, notably conditional GAN [Mirza and Osindero, 2014] and InfoGAN [Chen *et al.*, 2016] respectively. Here we demonstrate that our approach naturally offers an unsupervised solution for generating samples with constraints. Specifically, assume we have m data distributions (supposed to belong to m classes, although we do not need know these class labels explicitly). The goal is to generate samples for just one particular primary data class, and being as different from remainder data classes as possible. For example, we have an *unlabeled* primary data source including images of people with blond, black hairs and wish to generate only images with blond hair; we can find another data source containing images of people with black hair and use it as an auxiliary data source.

We conduct extensive experiments to demonstrate the merits of our simple yet very effective approach. We used the CIFAR-10, STL-10, and ImageNet datasets and computed Fréchet Inception Distance (FID) [Heusel *et al.*, 2017] to evaluate our solution to the mode collapsing problem against the baselines. The results show that our approach achieves the best FID scores on these real-world datasets and can generate high-quality images. Beyond addressing the mode collapsing problem, we further demonstrate that our framework is capable of learning from negative examples, e.g., learning to generate faces with non-black hairs while given example of faces with black hair.

2 Push and Pull GAN

We now describe how our P2GAN works. Recall that P2GAN maintains an existing set of generators that are currently well occupying some data modes and the principle for sequentially adding a new generator is to encourage this generator seeking for new missing data modes in order to boost the diversity. Guided by this principle, we propose using push forces to push the generated distribution Q of the new generator away those of the previous generators, i.e. $P_1, \ldots, P_m$, while using a pull force to pull the generated distribution Q towards the real data distribution P. We term our model *Push-and-Pull GAN* (P2GAN) and explicitly characterize the pull and push forces as f-divergences. Intuitively, since the previous generators can well occupy some data modes and the new generator is encouraged to generate data samples that mimic the true ones and diverge from the existing ones, this is expected to well explore and occupy some additional missing data modes. In what follows, we present the formulation of P2GAN when adding a new generator based on the existing ones, followed by some theoretical results enabling training P2GAN and a concise discussion on how to generalize P2GAN for other applications, specifically generating images with constraints. The supplementary material for this paper can be found at the following url address[1].

2.1 P2GAN Formulation

Let f_i(s) and ϕ be convex, lower semi-continuous functions. At each incremental round, our proposed P2GAN solves the following optimization problem to find a new generator Q

$$\max_{Q \in \mathcal{Q}} \left\{ \sum_{i=1}^{m} \alpha_i D_{f_i}\left(P_i \| Q\right) - D_{\phi}\left(P \| Q\right) \right\} \qquad (1)$$

where $\alpha_1, ..., \alpha_m \geq 0$ are the push parameters and $\mathcal{Q}$ is a suitable class of functions.

This formulation is natural as it is clear that the optimal solution Q^* for Eq. (1) is the closest to the data distribution P while furthermost from P_i (s) in the f-divergence optimization sense. The parameter α_i is a hyper-parameter to adjust how favorable we would like Q^* is to be different from P_i (s). Setting $\alpha_1, \ldots, \alpha_m$ to be small will favor Q^* to be closer to

[1] https://app.box.com/v/p2gan-supp.

P and vice versa[2]. In its most general form, the formulation allows us to use different kind of f-divergence for different pull and push forces. In our experiments, we will simply use standard JS for all divergences.

2.2 Training P2GAN

We now assume that the generative distribution Q is formed by a NN-based generator G and the source of randomness $\boldsymbol{z} \sim P_{\boldsymbol{z}}$ (i.e., the noise distribution). Let $S(\boldsymbol{x})$ be the *primary discriminator* specifying a score for $\boldsymbol{x}$ to be more likely generated from the primary distribution P rather than the generative distribution Q. Likewise, for each auxiliary distribution P_i, denote by $\tilde{S}_i(\boldsymbol{x})$ the *auxiliary discriminator* scoring the degree to which $\boldsymbol{x}$ is generated from P_i rather than the generative distribution Q. Once again, all discriminator functions S and $\tilde{S}_i$ (s) are parameterized by deep NNs.

We propose to solve the following optimization problem, which is later on proved in Theorem 2 to be equivalent to the our formulation in Eq (1):

$$\max_{G,\tilde{S}_{1:m}} \min_{S} \mathcal{J}\left(G, S, \tilde{S}_{1:m}\right) \tag{2}$$

where we have defined the objective function $\mathcal{J}\left(G, S, \tilde{S}_{1:m}\right)$ as:

$$-\mathop{\mathbb{E}}_{P}\left[g\left(S\left(\boldsymbol{x}\right)\right)\right] + \mathop{\mathbb{E}}_{P_{\boldsymbol{z}}}\left[\phi^*\left(g\left(S\left(G\left(\boldsymbol{z}\right)\right)\right)\right)\right]$$
$$+\sum_{i=1}^{m}\alpha_i\left[\mathop{\mathbb{E}}_{P_i}\left[\tilde{g}_i\left(\tilde{S}_i\left(\boldsymbol{x}\right)\right)\right] - \mathop{\mathbb{E}}_{P_{\boldsymbol{z}}}\left[f_i^*\left(\tilde{g}_i\left(\tilde{S}_i\left(G\left(\boldsymbol{z}\right)\right)\right)\right)\right]\right]$$

and $g, \tilde{g}_i$ (s) are the monotonic increasing wrapping functions mapping from $\mathbb{R}$ to $\mathrm{dom}\left(\phi^*\right)$ and $\mathrm{dom}\left(f_i^*\right)$ respectively to ensure a valid optimization problem.

We note that with monotonic increasing functions for Fenchel conjugate function in the f-divergences (e.g., as listed in our supplementary material), minimizing the first two terms in above objective function w.r.t S is expected to return high score $S(\boldsymbol{x})$ for $\boldsymbol{x} \sim P$ and low for $\boldsymbol{x} \sim Q$. Likewise, maximizing the last two terms w.r.t $\tilde{S}_i$ returns high score $\tilde{S}_i(\boldsymbol{x})$ for $\boldsymbol{x} \sim P_i$ and low score for $\boldsymbol{x} \sim Q$.

Building upon the result from [Nowozin *et al.*, 2016] for the single distribution case, we formally show in Theorem 1 and 2 that optimal solutions $\tilde{S}^*_{1:m}(\boldsymbol{x})$ and $S^*(\boldsymbol{x})$ for Eq. (2) when Q is held fixed can be obtained analytically, and that optimizing Eq. (2) is indeed equivalent to solving the optimization problem in Eq. (3). From a high level perspective, this results are natural, given the connection between f-divergence and a suitable optimal loss of the best classifier trying to separate data coming from the two distributions. One thing to remark about Eq. (2) is that despite having to deal with a set of new discriminators for the push forces, these discriminators only appear in the outer $\max$ problem, unlike the discriminator for the pull force which has to appear in the inner $\min$ problem. The technical details of all proofs can be found in the supplementary material[1].

Theorem 1. *Given the generative distribution Q (i.e., G), the optimal solutions $\tilde{S}^*_{1:m}(\boldsymbol{x})$and $S^*(\boldsymbol{x})$ of the optimization problem in Eq. (2) can be evaluated as*

$$\tilde{S}^*_i\left(\boldsymbol{x}\right) = \tilde{g}_i^{-1}\left(\nabla f_i\left(\frac{p_i\left(\boldsymbol{x}\right)}{q\left(\boldsymbol{x}\right)}\right)\right),\ 1 \le i \le m$$
$$\text{and } S\left(\boldsymbol{x}\right) = g^{-1}\left(\nabla\phi\left(\frac{p\left(\boldsymbol{x}\right)}{q\left(\boldsymbol{x}\right)}\right)\right)$$

Theorem 2 further establishes the Nash equilibrium point of the minimax problem in Eq. (2) to be a solution of the optimization problem in Eq. (3).

Theorem 2. *The optimization problem in Eq. (2) is equivalent to the following optimization problem:*

$$max_Q\left(\sum_{i=1}^{m}\alpha_i D_{f_i}\left(P_i \| Q\right) - D_\phi\left(P \| Q\right)\right) \tag{3}$$

2.3 Generating Data with Constraints

We can adopt the framework of P2GAN to tackle the problem of generating samples with constraints in an unsupervised manner. To be more precise, assume that we have a primary *unlabeled* data source with m classes: $C_1, \ldots, C_m$. We wish to train a generative model that can generate samples that mimic real data in the primary data source except those in some classes including $C_{i_1}, \ldots, C_{i_k}$. We further assume that we can find additionally auxiliary data sources of classes $C_{i_1}, \ldots, C_{i_k}$. We train a generative model generated data samples satisfying the aforementioned constraints by pushing from the auxiliary data sources while pulling to the primary data source. In addition, the auxiliary data sources do not need to be a part of the primary data source. For example, assume that we have a primary data source including images of people with black and blond hairs and we wish to generate only images with blond hair; we can find images from another data source which contains images of people with black hair and use it as auxiliary data source.

2.4 Addressing the Mode Collapse

The underlying idea of using the auxiliary data sources to address the mode collapse is as follows. We train the generator G_1 by minimizing $D_{f_1}\left(P \| Q_{G_1}\right)$ where $Q_{G_1}$is the push-forward distribution of the noise distribution via the generator G_1. It is likely that Q_{G_1} can only cover some modes in the real data distribution P. To enable the discovery of other modes in the real data distribution P, we set Q_{G_1} as the first auxiliary data distribution and train the second generator G_2 in such a way that its push-forward distribution Q_{G_2} minimizes $-D_h\left(Q_{G_1} \| Q_{G_2}\right) + D_{f_2}\left(P \| Q_{G_2}\right)$. This minimization encourages the resulting distribution Q_{G_2} to simultaneously move away the distribution Q_{G_1} and toward the real data distribution P, hence pushing Q_{G_2} to cover more other modes in the real data distribution P. This process is proceeded by setting Q_{G_1}, Q_{G_2} as two auxiliary data distributions and until the resulting distributions $Q_{G_1}, Q_{G_2}, \ldots, Q_{G_K}$ can cover most of modes in the real data distribution P.

[2]Note that we purposely do not specify a hyper-parameter for the pulling term $-D_\phi\left(P \| Q\right)$ as it is implicitly controlled via all α_i (s). However, in our discussion for a extended version of Eq. (1) later in the supplementary material, such parameters will need to be explicitly introduced.

3 Experiment

In this section, we first extensively demonstrate how our P2GAN can be used to address the mode collapsing problem, and achieving the best FID scores on CIFAR-10, STL-10, and ImageNet in comparison with current strongest baselines. This is then followed by another application to generate samples with constraints to demonstrate the flexibility of our P2GAN beyond its use for addressing the mode collapse. Regarding network architectures used in all experiments, the generators share parameters in all layers except for the weights from the input to the first hidden layer; the discriminators share parameters in all layers except for the weights from the penultimate hidden layer to the output layer; auxiliary sources generate the same pushing force, all α_i in Eq. (1) are equal, and their sum is denoted by α. We refer to the supplementary material for more details about model architectures and hyper-parameter setting.

3.1 Addressing the Mode Collapse with P2GAN

We now demonstrate the performance of our proposed model in addressing the mode collapsing problem on both synthetic 2D and real-world large-scale datasets. We compare the results of our method with those of the state-of-the-art GAN's variants by replicating experimental settings in the original work.

Synthetic Data

First, we reuse the experimental design proposed in [Metz *et al.*, 2016] to investigate how our P2GAN explores multiple data modes. The training data is sampled from a 2D mixture of 8 isotropic Gaussian distributions with a covariance matrix of $0.02\boldsymbol{I}$ and means arranged in a circle of zero centroid and radius of 2.0. The small variance creates low density regions, thus separating the modes.

We start with one generator and sequentially add a new generator after every $1,000$ training epochs until reaching 8 generators, and then train the entire network for $15,000$ epochs in total. The generators have an input layer with 256 noise units drawn from isotropic multivariate Gaussian distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$, and two hidden layers with 128 ReLU units each. The discriminators contain one hidden layer with 128 ReLU units. The pushing parameter α is set to 0.05. Fig. 1 shows the evolution of data generated by P2GAN. It can be seen that each new generator at first generates a cluster at the center of the circle and gradually moves to one of the unoccupied modes. Eventually at epoch $8,000$, each generator captures one mode, and 8 generators altogether effectively cover all the modes.

Real-world Datasets

In this section, we present our experiments on real-world datasets. We first conduct experiment on the CIFAR-10 dataset [Krizhevsky and Hinton, 2009] to investigate the influence of the number of generators. The result shows that P2GAN model helps stabilize training and improve visual quality of generated samples over the standard GAN with a similar architecture. Next we perform experiments on the STL-10 [Coates *et al.*, 2011] and ImageNet [Russakovsky *et al.*, 2015] datasets to prove that P2GAN not only achieves the best quantitative results but also generates highly realistic images.

Data and Evaluation Metric. CIFAR-10 contains 50,000 32×32 training images of 10 classes, including *airplane*, *automobile*, *bird*, *cat*, *deer*, *dog*, *frog*, *horse*, *ship*, and *truck*. STL-10 contains about 100,000 96×96 images, subsampled from ImageNet, and is a more diverse dataset than CIFAR-10. ImageNet is the largest and most diverse datasets with more than 1.2 million images from 1,000 classes. We follow the procedure of [Krizhevsky *et al.*, 2012] to resize the STL-10 and ImageNet images down to 48×48 and 32×32, respectively, for a fair comparison with the baselines in [Warde-Farley and Bengio, 2016; Miyato *et al.*, 2018; Hoang *et al.*, 2018]. We also resize ImageNet images down to 64×64 to examine P2GAN's capability on higher-resolution data. To quantitatively assess the quality and diversity of generated samples, we adopt Fréchet Inception Distance (FID) proposed in [Heusel *et al.*, 2017] that is more advanced than Inception score [Salimans *et al.*, 2016] since it compares the statistics of synthetic samples with those of the real samples, hence capturing the similarity of the two distributions better [Heusel *et al.*, 2017]. FIDs are computed on samples of 50,000 images.

Model Architecture with CNN and ResNet. Our network is designed following the DCGAN's architecture [Radford *et al.*, 2015], which we refers to as *Standard CNN*, and the *ResNet* architecture used in [Gulrajani *et al.*, 2017]. For the pulling force, we use the *Jensen-Shannon* (JS) divergence as in the standard GAN. For the pushing force, we experiment with both JS and KL divergences. The results are similar but the JS is more numerically stable, hence we eventually use it for both pulling and pushing. Discriminators of the same type, either *pull* or *push*, share parameters in all layers except for the weights from the last layer to the output layer. Starting with one generator, we add a new generator every fixed number of epochs until the number of generators reaches a predefined number K , and continue training the entire network. The learning process is terminated after 150 epochs for CIFAR-10, 100 epochs for STL-10, and 50 epochs for ImageNet.

Hyperparameter Setting. We use Adam optimizer with a batch size of 64. The learning rate and the first-order momentum are set at 0.0002 and 0.5, respectively. Regarding the pushing parameter α, we observe that varying α between 0.001 and 0.1 makes only little influence on the quantitative results, but large α can lead to less visually appealing samples as generators push each other too hard. As a result, we employed a gentle force of 0.01 for all experiment. We vary the total number of generators K in $\{1, 3, 5, 10, 15\}$ for our experiment on CIFAR-10 to investigate the influence of K. We add a new generator for every 15, 10, 5, 3 epochs when K is $3, 5, 10,$ and 15, respectively. For STL-10 and ImageNet, we simply set K at 10. For models using the ResNet architecture, we apply noise-reduced regularization [Roth *et al.*, 2017] and set the regularization parameter γ at 2.0.

The Influence of the Number of Generators. Tab. 1 compares FIDs (lower is better) for P2GAN with different values of K, the number of generators. All models use the ResNet

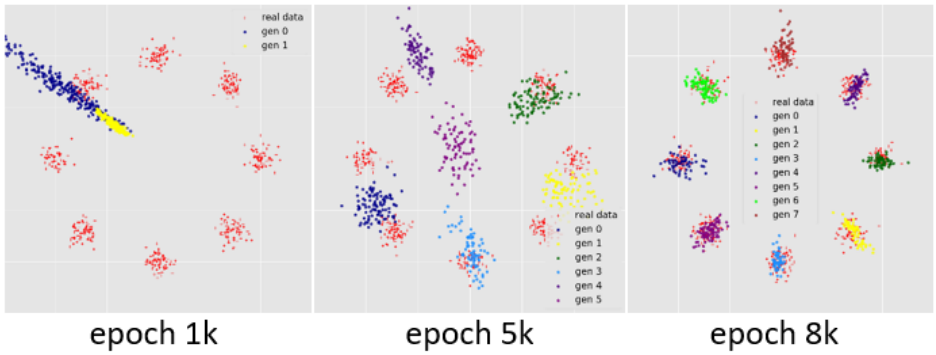
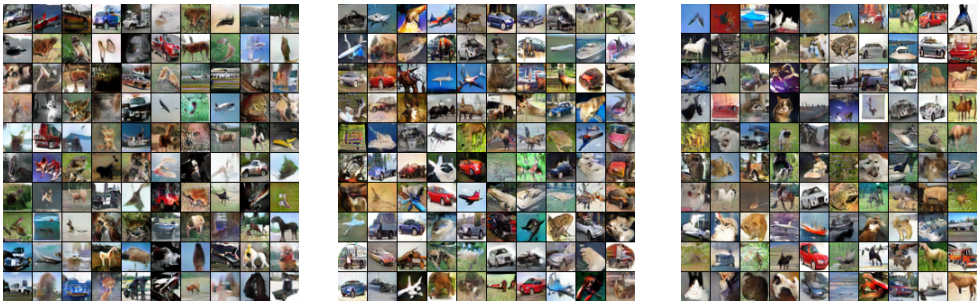


Figure 1: Evolution of data generated by P2GAN. Data samples from the 8 Gaussians are in red, and generated data by each generator are in a different color.



(a) Standard GAN. (b) First gen of 5-gen P2GAN. (c) 5-gen P2GAN.

Figure 2: Random samples generated by GAN models trained on CIFAR-10.

architecture. It should be noted that the P2GAN model with only one generator turns into the standard GAN, so we refer to it as *standard GAN*. The results show that using more generators improves FID and the performance peaks at $K = 5$. However, FID slightly deteriorates when K is 10 or 15. This behavior is consistent with that in our experiment with synthetic data (see Sec. 3.1) when the generators are crowded. In general, more diverse data can accommodate more generators.

# Generators	FID
1	26.7
3	25.7
5	**20.1**
10	23.1
15	23.5

Table 1: Comparison of FIDs (lower is better) on CIFAR-10 for P2GAN models with different number of generators.

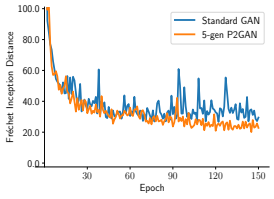


Figure 3: Comparison of FIDs for the P2GAN models with 1 and 5 generators on CIFAR-10. The 5-gen model is better and more stable FIDs.

We further compares the standard GAN model with the 5-gen P2GAN model. Fig. 3 plots FIDs of each model over training epochs. The 5-gen P2GAN achieves better and more stable performance. After 60 epochs, the FID of standard GAN becomes unstable and deteriorates. On the contrary, the FID of 5-gen P2GAN remains stable and keeps improving. Fig. 2 shows random samples generated by the standard GAN (a), the first generator of the 5-gen P2GAN (b) and the 5-gen P2GAN (c). Due to limited space, we refer to the supplementary material for samples generated by other generators of the 5-gen P2GAN. Compared to the standard GAN, the first generator of 5-gen P2GAN generates fewer classes of objects (mostly car, airplanes and birds) but much clearer images. As a result, the 5-gen P2GAN (i.e., P2GAN with 5 generators) produces more diverse and visually appealing samples. This analysis demonstrates that P2GAN has more stable learning, achieves stronger quantitative evaluation and generates better samples.

Fréchet Inception Distance Results. Tab. 2 reports the FIDs obtained by our P2GAN and the latest baselines collected from recent work in literature [Heusel *et al.*, 2017; Miyato *et al.*, 2018]. For fair comparison, we also implement MGAN [Hoang *et al.*, 2018] using the ResNet architecture with noise-reduced regularization. It is worthy to note that in the *standard CNN* group, DCGAN+TTUR and P2GAN share the same architecture similar to DCGAN, while WGAN-GP and SN-GANs employ a similar architecture for the generator, but add three more layers to the discriminator [Miyato *et al.*, 2018]. Overall, the P2GAN significantly outperforms other baselines on both CIFAR-10 and STL-10 in terms of FID. On the ImageNet dataset, our P2GAN model with ResNet architecture achieves a FID of *18.1* while MGAN with ResNet obtains *21.8*. Note that we have not found the FIDs of other baselines for this dataset. These impressive results demonstrate the effectiveness of P2GAN in addressing the mode collapse.

Generated samples. Random CIFAR-10 images generated by our proposed model with ResNet architecture were discussed and presented previously in Fig. 2. For 48×48 STL-

Model	CIFAR-10	STL-10
-Standard CNN-		
WGAN-GP	40.2	55.1
DCGAN [Radford *et al.*, 2015]	37.7	–
DCGAN + TTUR [Heusel *et al.*, 2017]	36.9	–
LayerNorm [Miyato *et al.*, 2018]	33.9	75.6
WeightNorm [Miyato *et al.*, 2018]	34.7	73.4
Orthonormal [Miyato *et al.*, 2018]	29	46
WeightClipping [Miyato *et al.*, 2018]	42.6	64.2
MGAN [Hoang *et al.*, 2018]	26.7	–
SN-GANs [Miyato *et al.*, 2018]	25.5	43.2
P2GAN (ours)	25.2	53.3
-ResNet-		
WGAN-GP [Gulrajani *et al.*, 2017]	29.3	–
WGAN-GP + TTUR [Heusel *et al.*, 2017]	24.8	–
SN-GANs [Miyato *et al.*, 2018]	21.7	40.1
MGAN [Hoang *et al.*, 2018]	20.9	45.3
P2GAN (ours)	**20.1**	**38.2**

Table 2: Comparison of FIDs (lower is better) with *unsupervised* image generators.

(a) SN-GAN. (b) P2GAN

Figure 4: Random 48×48 STL-10 samples generated by SN-GAN (left) and P2GAN (right).

10, Fig. 4 compares random samples generated by P2GAN and the closest baseline, SN-GAN [Miyato *et al.*, 2018]. SN-GAN samples sketch the general shapes of some objects but are blurry and fragmented. P2GAN samples are much sharper, more vivid and visually appealing depiction of a variety of objects. Lastly, we present highly recognizable 64×64 ImageNet samples and shows random 32×32 ImageNet samples generated by DFM, MGAN and P2GAN in the supplementary material[1]. These examples once again confirm the superiority of P2GAN's generated samples over those of other methods.

3.2 Generating Data with Constraints with P2GAN

In this experiment, we verify the effectiveness of the pushing force in our framework. We consider the scenario where the primary *unlabeled* data source consists of images from two classes, whilst the auxiliary *unlabeled* data source only has images from one of those two classes. The images in auxiliary source, though from the same class, are different from those contained in the primary data. In particular, we conduct the experiment where we have two classes: blond and black hair. Our task is to train generators to generate blond hair. In what follows we present our data construction, model setting and the results.

Figure 5: Samples generated by P2GAN model pushed by images of females with black hair. It can be seen that most generated images are of females with blond hair.

Data construction. For the blond/black hair setting, we randomly pick $12,000$ female images with black hair, and $12,000$ with blond hair from the CelebA dataset [Liu *et al.*, 2015], and merge them to create the primary data source. We then randomly select $11,000$ images of females with black hair, which do not appear in the primary data source, from the CelebA dataset to create the auxiliary data source.

Model setting. In generating data with constraints, the generator faces two conflicting goals. The first goal is to driven by the pulling force to generate data for both two classes. The second goal is driven by the pushing force to avoid generating data of the class in auxiliary source. To address this conflict, we train two generators G_1 and G_2 where: the primary source pulls both G_1 and G_2; the auxiliary source pulls G_2 but pushes G_1; and G_1 and G_2 push each other. The idea is that G_2 will cover a part of the primary data that is similar to the auxiliary data, and contribute additional force to push G_1 to generate images as our desired results. Here G_1 and G_2 also follow the same parameter sharing scheme as mentioned above.

Results. Empirically we observe that setting the pulling force produced by the primary data to 1.0 and applying a small pulling force of the auxiliary source of 0.1 and a gentle pushing force (of the auxiliary source, and between $G_1$and G_2) of 0.01 can effectively encourage G_1 and G_2 to generate different data. Fig. 5 shows samples generated by G_1 for blond/black hair experiment. It can be seen that most images generated by G_1 are of females with blond hair, whilst some tend to mix with other colors such as red, green or white.

4 Conclusion

In this paper, we enrich the formulations and applications of GANs by introducing push forces among generated samples and a heap of auxiliary data sources to propose Push-and-Pull GAN (P2GAN). We demonstrate two applications of P2GAN in generating data with constraints and addressing the mode collapsing problems. Besides these two applications, our P2GAN might be applicable to other applications such as augmenting data for anomaly detection problems; however, we leave these investigations to our future work.

References

[Arjovsky *et al.*, 2017] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[Chen *et al.*, 2016] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016.

[Coates *et al.*, 2011] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223, 2011.

[Dam *et al.*, 2019] Nhan Dam, Quan Hoang, Trung Le, Tu Dinh Nguyen, Hung Bui, and Dinh Phung. Three-player wasserstein gan via amortised duality. In *Proc. of the 28th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 2019.

[Goodfellow *et al.*, 2014] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[Goodfellow, 2016] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

[Gulrajani *et al.*, 2017] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5769–5779, 2017.

[Heusel *et al.*, 2017] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6629–6640, 2017.

[Hoang *et al.*, 2018] Quan Hoang, Tu Dinh Nguyen, Trung Le, and Dinh Phung. MGAN: Training generative adversarial nets with multiple generators. In *International Conference on Learning Representations*, 2018.

[Isola *et al.*, 2017] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint*, 2017.

[Krizhevsky and Hinton, 2009] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[Le *et al.*, 2018] Trung Le, Hung Vu, Tu Dinh Nguyen, and Dinh Phung. Geometric enclosing networks. In *Proc. of the 27th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 2018.

[Liu *et al.*, 2015] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3730–3738, 2015.

[Mathieu *et al.*, 2015] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015.

[Metz *et al.*, 2016] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*, 2016.

[Mirza and Osindero, 2014] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[Miyato *et al.*, 2018] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *International Conference on Learning Representations*, 2018.

[Nguyen *et al.*, 2017] Tu Dinh Nguyen, Trung Le, Hung Vu, and Dinh Phung. Dual discriminator generative adversarial nets. In *Advances in Neural Information Processing Systems 29 (NIPS)*, 2017.

[Nowozin *et al.*, 2016] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems*, pages 271–279, 2016.

[Radford *et al.*, 2015] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[Roth *et al.*, 2017] Kevin Roth, Aurelien Lucchi, Sebastian Nowozin, and Thomas Hofmann. Stabilizing training of generative adversarial networks through regularization. In *Advances in Neural Information Processing Systems 30*, pages 2018–2028. 2017.

[Russakovsky *et al.*, 2015] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[Salimans *et al.*, 2016] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

[Warde-Farley and Bengio, 2016] David Warde-Farley and Yoshua Bengio. Improving generative adversarial networks with denoising feature matching. 2016.